Amendments to the Specification

rewrite the paragraph bridging pages 1 and 2 as follows (text to be added is underlined; text to be deleted is struck-out):

The so-called "Slepian-Duguid" prior art algorithm has been used to schedule connections in rearrangeably non-blocking switches. This algorithm makes use of "Paull's matrix", which is formed by labelling the switch's input ports as the matrix rows; and by labelling the switch's output ports as the matrix columns. Each matrix entry (i, j) is either blank, indicating no connection between switch input port i and output port j; or, contains an identifier for the intermediate switch on which the connection of input port i to output port j is allocated. The basic Slepian-Duguid algorithm is depicted in flowchart form in Figure 1. Although the algorithm is shown with a failure terminal point ~~points~~, the failure ~~these failures~~ only indicates ~~indicate~~ that either the input port or output port has been over-allocated. The Slepian-Duguid algorithm always succeeds on loads of up to 100% capacity.

rewrite the paragraph bridging pages 4 and 5 as follows:

The algorithm then schedules connection (3,1) for switch A as indicated in Figure 1, block 20 and as illustrated in Figure 6. It is then necessary to reschedule or "flip" connection (2,1) from switch A to switch B to prevent over-allocation of switch A's connection to output 1. This is accomplished as shown in Figure 1, blocks 22-34. First, as indicated in block 22, the existing switch A entry (i.e. that for connection (2,1)) in column 1 is flipped to switch B (i.e. $i'=2$). A test (block 24) is then made to determine whether the Paull's matrix representation of the Figure 2 switch fabric includes another entry for switch B in row $i'=2$. The answer in this case is "yes", namely the previously scheduled connection (2,3). This necessitates flipping connection (2,3) from switch B to switch A to prevent over-allocation of the input connection to switch B. More particularly, if the block 24 test result is "yes", processing continues with block 26 such that the existing switch B entry (2,3) is flipped to switch A (i.e. $i'=2$ and $j'=3$). The algorithm then tests (block 30) the Paull's matrix representation of the Figure 2 switch fabric to determine whether there is another entry for switch A in column $j'=3$. If the answer is "no" the algorithm terminates successfully as indicated at block 34, but the ~~The~~ answer in this case is "yes", namely the previously scheduled connection (1,3). This necessitates flipping connection (1,3) from switch A to switch B to prevent over-allocation of the input connection to switch A. More particularly, if the block 30 test result is "yes", processing continues with block 32 which assigns $i=i'=2$ and $j=j'=3$. Accordingly, when processing then continues with block 22, the existing switch A entry in row 1, column 3 is flipped to switch B (i.e. $i'=1$). The block 24 test is then again made to determine whether the Paull's matrix representation of the Figure 2 switch fabric includes another entry for switch B in row 1. The answer in this case is "no", and the algorithm terminates successfully as indicated at block 28.

Rewrite the paragraph at page 9, lines 6-8 as follows:

Figure 12 depicts joinder of Figure 11 waves A and B to satisfy an additional request for connection (3,1) in accordance with the Figure 10 ~~12~~ join flow aspect of the present invention.

Rewrite the paragraph at page 11, lines 6-23 as follows:

Figures 9 and 10 show how a new connection (*c*, *d*) is added to a switch consisting of I, M and O, with the state of M described by S. During the primary flow process which starts at block 40 depicted in Figure 9, an attempt (block 42) is first made to find a wave in which both the input node and the output node have no attached edges. For example, in Figure 11, wave A has no edges attached to input node 3 and output node 2; and, wave B has no edges attached to input node 1 and output node 1. If such a wave in which both the input node and the output node have no attached edges is found, then the edge may be added (Figure 9, block 44) to the wave without any rearrangement of existing connections and the algorithm terminates successfully (block 46). If no wave having both input and output nodes free is found, then a further search (Figure 9 blocks 48, 52) is made to locate two waves, one with the input free and one with the output free, and these two waves are then joined (block 56) by the process depicted in Figure 10, blocks 60, 62, 64, 66, 68, 70 and 72. The algorithm then terminates successfully (Figure 9, block 58). Like the Slepian-Duguid algorithm, the invention never fails on a 100% or lighter load. The failure termination points (Figure 9 blocks 50, 54) are reached only if some input port or output port is over-allocated in a connection request.

Rewrite the paragraph at page 13, lines 1-10 as follows:

For example, suppose that a request for connection (3,1) arrives while the Figure 2 switch is supporting the connection schedules depicted in Figure 11. Figure 12 shows how the Figure 10 process joins Figure 11 waves A and B with connection (3,1) to satisfy the new connection request. First, as indicated in Figure 10, block 62 and as shown in the upper portion of Figure 12, waves A and B are joined by creating a single graph which includes all of the edges of waves A and B, with each edge labelled to identify the wave in which that edge originated. The new connection (3,1) is unlabelled in the upper portion of Figure 12.